VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Testování embedded databázových systémů

Benchmarking of Embedded Database Systems

2018 David Tabor

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **David Tabor**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Testování embedded databázových systémů
Benchmarking of Embedded Database Systems

Jazyk vypracování: čeština

Zásady pro vypracování:

Dnešní relační databázové systémy se dělí podle architektury na systémy typu klient-server a embedded systémy. Úkolem této práce je testování embedded databázových systémů z hlediska použitých datových struktur a indexů a jejich porovnání.

Bakalář musí v rámci bakalářské práce:
1. Nastudovat datové struktury implementované v softwarovém rámci RadegastDB vyvíjeném na Katedře informatiky.
2. Prostudovat datové struktury existujících embedded databázových systémů (SQLite, Embedded MySQL, ...) a porovnat jejich možnosti a výkon se softwarovým rámcem RadegastDB.
3. Vytvořit webové rozhraní pro vizualizaci testovaných databázových systémů.
4. Porovnat výkon datových struktur testovaných embedded databázových systémů a softwarového rámce RadegastDB.

Seznam doporučené odborné literatury:

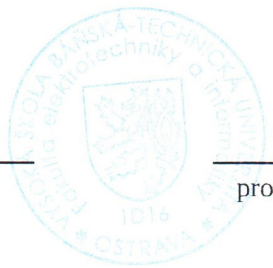
[1] Jay A. Kreibich. Using Sqlite (1st ed.). O'Reilly Media, Inc., 2010.
[2] Michael Olson, Keith Bostic, Margo Seltzer. Berkeley DB. Proceedings of the FREENIX Track: 1999 USENIX Annual Technical Conference

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Peter Chovanec, Ph.D.**

Datum zadání: 01.09.2016

Datum odevzdání: 30.04.2018



doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2018 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 30. dubna 2018 .....

Rád bych na tomto místě poděkoval vedoucímu bakalářské práce Ing. Petrovi Chovancovi, Ph. D. za odborné vedení a pomoc při realizaci této práce.

Abstrakt

Cílem této bakalářské práce je vytvořit univerzální testovací rozhraní pro měření výkonu embedded databázových systémů v závilosti na zvoleném indexu, čili datové struktuře. Testovací rozhraní bude navrženo tak, aby jej bylo možno jednoduchou implementací doplnit o další embedded databázové systémy. Testy budou probíhat nad zvolenými embedded databázovými systémy, zvolenou datovou strukturou a kolekcí dat, která bude generována nebo nahrána z externího souboru a budou uživatele informovat prodstřednictvím webového rozhraní o stavu a průběhu. Historie testů bude ukládána v lokální databázi pro následné vyhodnocení.

Klíčová slova: Embedded databázové systémy, RadegastDB, SQLite, MySQL, Firebird SQL, SQL CE, Sekvenční pole (Halda), B-strom, R-strom, Hašovací tabulka

Abstract

The aim of this bachelor thesis is to create a universal test interface for measuring the performance of embedded database systems depending on the selected index, i.e. the data structure. The test interface will be designed to be complemented with other embedded database systems by simple implementation. Tests will run over selected embedded database systems, a selected data structure and a data collection that will be generated or loaded from an external file and will inform users via the web interface about status and progress. The history of the tests will be stored in a local database for subsequent evaluation.

Key Words: Embedded database systems, RadegastDB, SQLite, MySQL, Firebird SQL, SQL CE, Sequential array (Heap), B-tree, R-tree, Hash table

Obsah

Seznam použitých zkratek a symbolů

SŘBD	–	Systém řízení báze dat
DB	–	Databáze
eDB	–	Embedded databáze
MOO	–	Minimální ohraničující objekt
OOP	–	Objektově Orientované Programování
ADO.NET	–	Softwarová komponenta pro SŘBD
EF	–	Entity Framework
JS	–	JavaScript
AJAX	–	Asynchronní JavaScript a XML
JSON	–	JavaScriptový objektový zápis
MVC	–	Architektura Model-View-Controller
WWW	–	World Wide Web
HTML	–	Značkovací jazyk pro tvorbu webových stránek
HTTP	–	Protokol pro přenos HTML obsahu

Seznam obrázků

Seznam tabulek

1 Úvod

V dnešní době se již každý z nás setkal s některým z databázových systémů, aniž by o tom měl vůbec tušení. Příkladem databáze může být kartotéka, telefonní seznam, účetnictví a další. Databázové systémy slouží k uložení dat a následné práci s nimi, ať už se jedná o složité výpočty nebo jejich procházení a vyhledávání v nich. Zvyšují efektivitu a rychlost práce s daty. To vše díky implementaci indexovaných struktur. Index je datová struktura, která slouží ke zvýšení výkonnosti vyhledávání dat. Existují různé typy indexů s odlišnými vlastnostmi a jejich využití závisí na charakteru dat a dotazování se nad nimi. Základní sadu indexů, neboli datových struktur, nalezneme níže.

Databázové systémy se dělí dle architektury na systémy klient-server a embedded systémy. Základní rozdíl mezi nimi je, že systémy klient-server běží nezávisle na aplikaci, která přistupuje k datům, kdežto embedded systém je součástí aplikace. Jeden z embedded databázových systému, nazývaný RadegastDB, je vyvíjen na Katedře informatiky, Vysoké školy Báňské-Technické univerzity Ostrava. A právě tímto typem databázových systémů se zabývá tato bakalářská práce.

Výhodou tohoto typu databázových systémů je

- **vysoký výkon** - odpadající režie klient-server SŘBD, lokální přístup
- **spolehlivost** - neposkytují zdaleka všechny možnosti klient-server SŘBD, z toho vyplývá jednodušší údržba kódu
- **vysoká škálovatelnost**

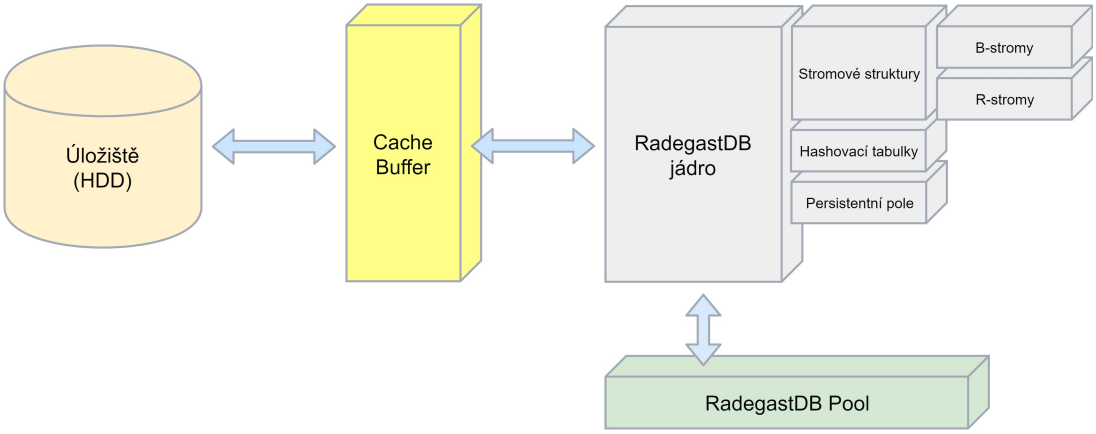
Mezi nevýhody se řadí

- **přístupová práva** - obvykle neimplementují žádné principy uživatelských práv a přístupu
- **zamykání** - většina eDB nepovoluje přístup více uživatelů v daný okamžik, případně při zápisu dochází k uzamčení celé databáze
- **podpora SQL** - zdaleka ne všechny embedded systémy implementují SQL dialekt

Cílem této bakalářské práce je nastudovat základní typy datových struktur a provést jejich porovnání v rámci různých embedded systémů. Pro porovnání bude sloužit testovací rozhraní, které bude zobrazovat průběh testování s následným vyhodnocením datové propustnosti a časové náročnosti.

2 Datové struktury RadegastDB

RadegastDB[2] je embedded databázový systém vyvíjený Katedrou informatiky, Fakulty elektrotechniky a informatiky, Vysoké školy Báňské-Technické univerzity Ostrava. Systém implementuje několik základních datových struktur jako je sekvenční pole (halda)[1], B-strom[3], R-strom[4] a hašovací tabulka[6]. Každá datová struktura podporuje, kromě základních operací jako vkládání, úpravy a mazání, i bodový či rozsahový dotaz. Systém také obsahuje vlastní mechanismus pro práci s ukládáním dat do mezipaměti či na disk, zamezující alokaci paměti bez žádosti, viz. architektura na obrázku č. 1. Na rozdíl od ostatních testovaných embedded databázových systémů neimplementuje SQL vrstvu, a proto práce s ním probíhá instancovaným objektem a množinou vyhrazených metod.



Obrázek 1: Architektura RadegastDB[2]

2.1 Sekvenční pole

Sekvenční pole[1] nebo-i halda je stránkované pole, ve kterém jsou data ukládána nejčastěji v pořadí, v jakém byla vkládána - nesetřízené pole, viz. obrázek č. 2. Složitost vkládání do nesetřízeného pole je $O(1)$ čili konstantní.

Vyhledávání v poli probíhá jeho sekvenčním průchodem od prvního prvku, dokud není nalezena shoda nebo nedojde-li na konec pole, viz. obrázek č. 2. Složitost tohoto vyhledávání je $O(n)$, kde n je velikost pole.

V SŘBD se sekvenční pole využívá k reprezentaci tabulky, přičemž jeden záznam představuje právě jeden prvek v poli.

Obrázek 2: Sekvenční pole - vkládání a vyhledávání[1]

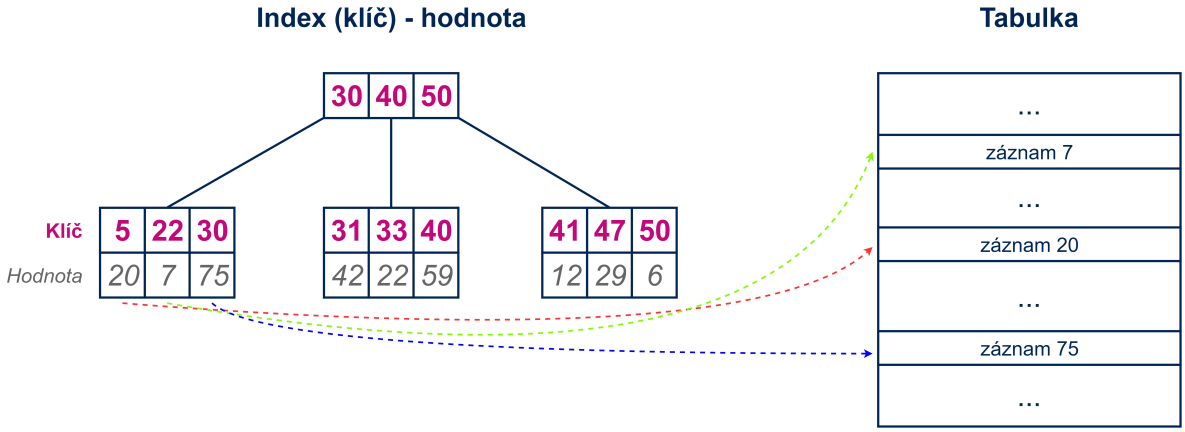
2.2 B-strom

B-strom[3] představuje velice efektivní stromovou strukturu pro uchovávání a vyhledávání hodnot, jež byla publikována roku 1972 autory Rudolf Bayer-em a Edward M. McCreight-em. Samotný B-strom se však v praxi neužívá. Nejužívanější je jeho modifikace B^+-strom. V současnosti se o B^+-stromu často hovoří jako o B-stromu.

B-strom řádu n je $(2n+1)$-arní výškově vyvážený strom, který splňuje následující kritéria:

1. kořen má alespoň 2 potomky, není-li listem
2. každý uzel obsahuje alespoň n klíčů a nejvýše $2n$ klíčů, s výjimkou kořene
3. každý uzel je buď listem (nemá žádné následníky) nebo má $(m+1)$ následovníků, kde m je počet klíčů v uzlu
4. všechny listové uzly jsou na stejné úrovni (výšková vyváženost)
5. data jsou umístěna pouze v listech

Datová struktura je tvořena dvojicí klíč-hodnota, kde hodnota je odkaz na záznam v tabulce, viz. obrázek č. 3.



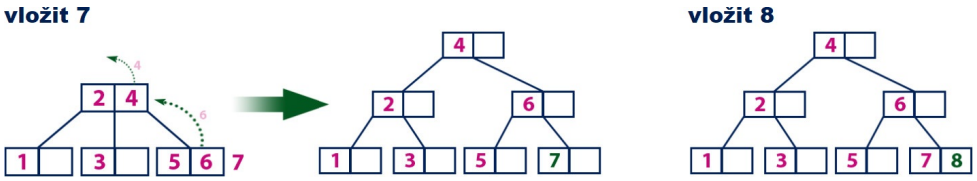
Obrázek 3: B-strom - klíč-hodnota[3]

Při vkládání je nejjednodušší případ, kdy se dá klíč přidat do uzlu, jenž ještě není zaplněn, tzn. neobsahuje $2n$ položek. V tomto případě se začlení nový klíč do uzlu při zachování uspořádání klíčů. V případě, že uzel je již naplněn, je nutné provést úpravy struktury stromu, které vedou k vytvoření jednoho nebo více nových uzlů.

1. mějme uzel U, který rozdělíme štěpením na dva uzly U a V
2. všech $2n$ klíčů se rozdělí rovnoměrně mezi tyto uzly, přičemž nám zůstane jeden klíč K nezařazen. Uzel U obsahuje klíče $k_i \leq K, 1 \leq i \leq n$ a uzel V $k_i \geq K, 1 \geq i \leq n$.

3. zbývá nám zařadit klíč K do předchůdce uzlu U, což znamená přidání nové položky do tohoto uzlu. V případě, že uzel je již naplněn, pokračujeme bodem 1.

Z tohoto postupu vyplývá, že nejhorší složitost vkládání je $O(\log_n(m))$, kde m je počet uzlů. Viz. obrázek č. 4.

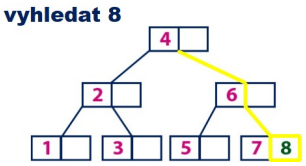


Obrázek 4: B-strom - vkládání[3]

Vyhledávání ve stromu probíhá následujícím způsobem. Klíče v uzlu jsou uspořádány od nejmenšího po největší. Označme si klíče $k_1, k_2, ..., k_m$. V případě, že hledaná hodnota klíče x se nerovná žádnému z klíčů k_i, pokračuje prohledávání uzlu následovníka dle následujících pravidel

1. jestliže $k_i < x < k_{i+1}$ pro $1 \leq i < m$, pokračujeme zpracováním uzlu následovníka p_i
2. jestliže $k_m < x$, vyhledávání pokračuje v uzlu p_m
3. jestliže $x < k_1$, vyhledávání pokračuje v uzlu p_0

Jestliže tímto výběrem následník není nalezen, položka s klíčem x ve stromu neexistuje. Složitost tohoto vyhledávání je $O(\log_n(m))$. Viz. obrázek č. 5.



Obrázek 5: B-strom - vyhledávání[3]

2.3 R-strom

R-strom[4] je prostorová datová struktura navržená Antoninem Guttmanem[5] v roce 1984. R-strom představuje prostorovou modifikaci B-stromu, kde záznamy v listových uzlech stromu obsahují odkazy na datové objekty reprezentující konkrétní prostorové objekty. Tato datová struktura využívá k indexaxi tzv. minimální ohraničující obdelníky (dále jen MOO). Obdobně jako u B-stromu, se samotný R-strom v praxi nevyužívá. Nejužívanější je jeho varianta R*-strom, o kterém se v současnosti často hovoří jako o R-stromu.

Vlastní indexace objektů je dána pomocí dvojic *(I, Id)*, přičemž I je MOO ohraničující odpovídající prostorový objekt. Dvojice *(I, Id)* jsou nazývány indexové záznamy. Každé I má obecně

tvar *(I_0, I_1, ..., I_{k-1})*, kde I_i je interval (a_i, b_i) popisující ohraničení objektu v dimenzi i. Pro $k = 2$ tedy potřebujeme 4 parametry. Nelistové uzly obsahují řídící záznamy tvaru $(I, ukazatel)$, kde ukazatel ukazuje na podstrom R-stromu takový, že I pokrývá všechny MOO, které se v něm vyskytují. U R-stromů není striktně dodrženo pravidlo o polovičním naplnění uzlu v nejhorším případě, jako je tomu u B-stromů. Je-li R-strom m-ární strom, pak m_1 bude označovat parametr, pro který platí $m_1 \leq m/2$. Někdy je R-stromu přiřazen řád (m_1, m). Minimální počet následníků uzlu R-stromu bude m_1.

R-strom řádu (m_1, m) je tedy m-ární strom, který má následující vlastnosti:

1. každý jeho nelistový uzel má n bezprostředních následníků, kde $n \subseteq \langle m_1, m \rangle$
2. každý listový uzel obsahuje n indexových záznamů $\subseteq \langle m_1, m \rangle$
3. kořen má nejméně dva bezprostřední následníky, pokud není listem
4. všechny cesty v R-stromech jsou stejně dlouhé

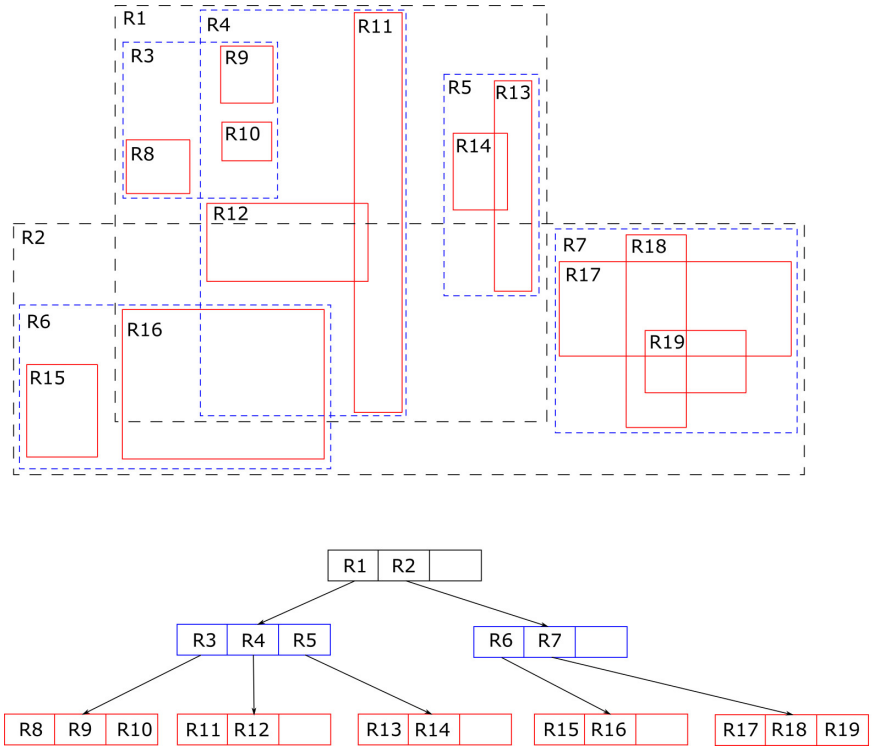
Pro x indexových záznamů je třeba v nejhorším případě hloubka R-stromu $(\log_m x) - 1$, nejhorší naplněnost stránky je m_1/m.

Při vkládání nového prostorového objektu může dojít k tomu, že jeden z uzlů odpovídajícího R-stromu, jenž má právě m odkazů, přesáhne svou kapacitu. V tomto případě dochází ke štěpení stránky, tedy MOO odpovídající danému uzlu se nahradí dvěma MOO. Analogicky pro uzel, který má právě m_1 odkazů, při rušení prostorového objektu, klesne počet odkazů pod m_1. V takovém případě musí následovat akce slévání stránek odpovídající slučování MOO. Obě tyto akce se mohou rekurzivně šířit do kořenového uzlu.

Na rozdíl od B-stromů není hledání v R-stromu určeno jednou větví. Protože MOO v jednotlivých podstromech se mohou překrývat, je možné, že existuje více než jedna možnost, jak pokračovat při prohledávání stromu z jednoho uzlu, viz. obrázek č. 6. Tím je hledání složitější a veškeré optimalizace používané při konstrukci R-stromu jsou založeny na požadavku, co nejvíce separovat MOO, aby se omezil prostor vyhledávání. Složitost vyhledávání v R-stromu je $O(log_m n)$. Z toho plynou speciální požadavky na algoritmus operace vkládání, ve kterém se realizují různé heuristiky nabízející víceméně uspokojivá řešení daného problému.

Algoritmus vkládání indexového záznamu do R-stromu má tedy pro následné zpracování dotazů zásadní důležitost. Je založen na strategii nalézt takovou větev ve stromě, tj. takový list R-stromu, že opravy obdélníků po cestě ke kořenu povedou k co nejmenším změnám. Důležitá bude i procedura pro štěpení. Tam se řeší problém, jak rozdělit neuspořádanou množinu obdélníků do dvou uzlů. Rozdělení záznamů do dvou uzlů je děláno tak, aby bylo co nejméně pravděpodobné, že bude potřeba oba uzly při prohledávání zkoušet. Protože uzly jsou navštěvovány z uzlu-předchůdce, je nutné minimalizovat objem odpovídající oblasti I. Pro rozdělení obsahu uzlů je možné použít algoritmus uvažující všechny možnosti. Hledá se globální minimum,

přičemž algoritmus má exponenciální složitost. Guttman uvádí další algoritmy, lineární a kvadratický, které pouze aproximují řešení.



Obrázek 6: Základní architektura R-stromu[4]

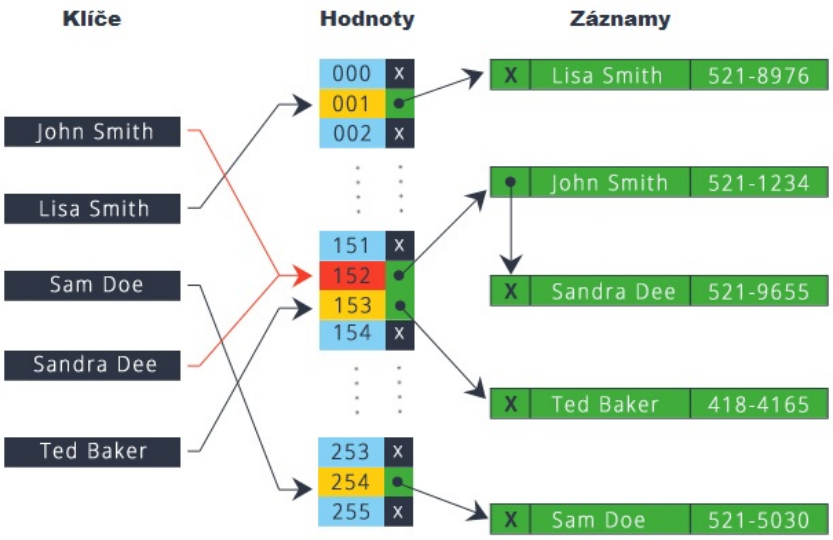
2.4 Hašovací tabulka

Hašovací tabulka[6] je datová struktura tvořena pomocí dvojice klíč-hodnota, kdy klíč je vypočítáván na základě hodnoty, pomocí některého z hašovacích algoritmů.

Při vkládání je nejdříve hašovací funkcí vytvořen index (klíč) a jeho hodnota je obvykle odkaz na uložený záznam v tabulce. Složitost uložení je konstantní $O(1)$. Může však dojít ke kolizi, kdy daný index je již obsazen. Tato kolize může být řešena dvěma způsoby - zřetězené rozptylování (separate chaining) nebo otevřené rozptylování (open addressing), viz. obr. č. 7.

Při použití metody zřetězeného rozptylování se na daném indexu vytváří spojový seznam a nový prvek je umístěn vždy na konec seznamu. Při nesprávném použití může dojít k degradaci tabulky na několik spojových seznamů. S použitím metody otevřeného rozptylování je na daném indexu připraveno pole s pevnou délkou a nový prvek je umístěn na první volnou pozici. U této metody může opět dojít ke kolizi, kdy pole nebude mít dostatečnou délku a nový záznam tedy nelze uložit. Tuto kolizi lze řešit pomocí strategie Linear probing[7] nebo Double hashing[8].

Díky oběma rozptylovacím metodám řešení kolize se při vyhledávání kombinují výhody vyhledávání pomocí indexu s konstantní složitostí $O(1)$ a procházení seznamu se složitostí $O(n)$.



Obrázek 7: Hašovací tabulka - klíč-hodnota[6]

2.5 Souhrn

V této kapitole byl popsán databázový framework RadegastDB a základní datové struktury, které poskytuje a jsou používané i v jiných databázových systémech. V následující kapitole popíšeme existující embedded databázové systémy a datové struktury, které poskytují a budou součástí srovnávacích testů testovacího rozhraní.

3 Embedded databázové systémy

Dnešní databázové systémy se dělí dle architekruty na systémy klient-server a embedded systémy. Databázové systémy klient-server jako Microsoft SQL Server[17], Oracle[9] nebo MySQL[13] fungují nezávisle na aplikacích, které je využívají. Běží v nezávislém procesu, často na jiném stroji, a aplikací komunikují prostřednictvím definovaného rozhraní, typicky např. ADO.NET[10] a dialekt SQL.

Embedded databázové systémy jsou tomu opakem. Jsou součástí námi vyvíjené aplikace, od toho označení embedded. Využívají stejných prostředků jako naše aplikace (paměť, úložiště, atd.), ale i navzdory tomu embedded databázové systémy nabízejí vysoký výkon, spolehlivost a škálovatelnost. Co však často nenabízí, jsou rozšířené funkce klient-server databázových systémů jako jsou přístup více uživatelů, procedurální nadstavba, replikace, zálohování, apod. Jsou tedy vhodné především v aplikacích, kde se využije uchování strukturovaných dat a rychlost práce s nimi. Pro chod embedded databázového systému často stačí jen referovat správné knihovny.

Embedded databázové systémy se často, pro svou jednoduchost, nenáročnost a dostupnost, využívají i v mobilních aplikacích. Právě těmito embedded systémy se zabývá tato bakalářská práce a proto se podíváme na vybrané systémy blíže v této kapitole.

3.1 Firebird

Firebird[11] nebo také FirebirdSQL vznikl jako alternativa open source databáze InterBase[12] po jejím uvolnění společností Borland v roce 2000. Od verze Firebird 1.5 byl však kód z velké části přepsán a jako open source projekt je nyní vyvíjen skupinou Firebird Project, kterou tvoří především rusky mluvící komunita vývojářů. Firebird v současné době nabízí verze 2.5, 3.0 a pro testování byl uvolněn alpha release verze 4.0. Jazyk použitý k implementaci je C++. Firebird implementuje datové struktury sekvenční pole a B-strom.

Systém je možné užívat jako instalaci klient-server databázového systému nebo jako embedded systém a jedná se o open source. Dále nabízí 32bit i 64bit verzi a je multiplatformní. Je možné jej provozovat na systémech Linux, Microsoft Windows i Mac OS.

3.2 MySQL

MySQL[13] byl vytvořen švédskou společností MySQL AB v roce 1995. Hlavními tvůrci byli David Axmark, Allan Larsson a Michael Wedenius. V roce 2008 společnost Sun Microsystems převzala společnost MySQL AB. Tu v roce 2010 převzala společnost Oracle Corporation. MySQL je od počátku nabízen ve variantách jako open source (nekomerční), ale i jako licencovaný software (komerční). Nejnovější verze k dispozici je 5.7 a pracuje se na verzi 8.0. Jazyk použitý k implementaci je C, C++. MySQL implementuje datové struktury sekvenční pole, B-strom a hašovací tabulka.

Dostupná verze obsahuje i knihovnu pro embedded implementaci[14], která však má být ve verzi 8.0 odstraněna. MySQL je multiplatformní s podporou 32bit i 64bit operačních systémů Linux, Microsoft Windows, MAC OS a dalších.

Zajímavostí je, že v roce 2010 vznikla z MySQL 5.5 alternativa MariaDB[15], kterou vytvořili právě jeho původní tvůrci. Mezi významné uživatele této alternativy patří Wikipedia[44], Wordpress.com[45] a Google[46].

3.3 Microsoft SQL Server Compact Edition

Microsoft SQL Server Compact Edition[16] nebo také SQL CE je odlehčená embedded verze SQL Serveru[17] společnosti Microsoft. První verze byla vydána spolu s verzí SQL Serveru 2005 v roce 2005. Poslední verze 4.0 byla vydána v roce 2011 a nyní už je společností Microsoft označena jako zastaralá. Jazyk použitý k implementaci byl C++. SQL CE implementuje datové struktury sekvenční pole a B-strom.

Platformně je omezen pouze pro operační systémy Microsoft Windows s podporou 32bit a 64bit verze.

3.4 SQLite

Autor Dwayne Richard Hipp dal vznik první verzi SQLite[18] v roce 2000. Ve verzi 1.0 využíval jako systém úložiště gdbm[19], který od verze 2.0 nahradil vlastní implementací B-stromu. Současně je k dispozici verze 3.0 a jako jazyk implementace je použit jazyk C. SQLite implementuje datové struktury sekvenční pole, B-strom a R-strom.

SQLite je možno provozovat na systémech Linux, Microsoft Windows, MAC OS, Android a mnoha dalších ve verzi 32bit i 64bit. SQLite se stal jedním z nejoblíbenějších embedded databázových systémů pro jeho výkon, spolehlivost a jednoduchost implementace.

Mezi významné uživatele patří např.:

- webové prohlížeče např. Google Chrome[47], Mozilla Firefox[48], Opera[49] či Android prohlížeč[56]
- frameworky webových aplikací Django[50], Drupal[51], Ruby on Rails[52] a další
- operační systémy jako Blackberry 10 OS[53], Symbia[54], Nokia Maemo[55], Android[56], LG webOS[57], NetBDS[58], Apple iOS[59], Windows 10[60]

Implementace je i součástí Microsoft Entity Framework Core[20] a nahrazuje tak SQL CE.

3.5 Berkeley DB

Berkeley DB[21] byl vyvinut na Kalifornské univerzitě v Berkeley jako databový systém typu klíč-hodnota a původně byl součástí operačního systému BSD, což byl systém na bázi UNIXu vyvinut na téže univerzitě. Poprvé byl systém Berkeley DB samostatně spuštěn roku 1991 a později byl opět zahrnut do systému BSD 4.4[61]. V roce 1996 společnost Netscape požádala autory, aby Berkeley DB rozšířili a specifikace tak odpovídala požadavkům pro LDAP[62] server a prohlížeč Netscape. Tato žádost vedla k založení společnosti SleepyCat Software, kterou v roce 2006 získala společnost Oracle. Ta se nyní stará o další vývoj Berkeley DB.

Berkeley DB je nyní rodinou multiplatformních embedded systémů s implementací v jazyce C (Berkeley DB), v jazyce Java (Berkeley DB JE) a v jazyce C++ (Berkeley DB XML). Berkeley DB implementuje datové struktury sekvenční pole, B-strom, hašovací tabulka.

Berkeley DB poskytuje základní systém ukládání a vyhledávání několika serverů LDAP, databázových systémů a mnoha dalších proprietárních aplikací a aplikací s open source kódem. Mezi významné uživatele patří např.:

- 389 Directory Server[63] - open source LDAP server od vývojářů Fedora Project
- BitCoin Core[64] - první implementace kryptoměny BitCoin
- Subversion[65] - centralizovaný systém pro správu a verzování kódu
- dále antispamové systémy Bogofilter[66], Spamassassin[67] a nosql[68] databáze Oracle NoSQL[69] a Voldemort[70]

3.6 eXtremeDB, Perst

McObject představilo eXtremeDB[22] v roce 2001. Systém je zaměřený především na embedded řešení aplikací běžících v prostředí s omezeným přístupem k prostředkům. Mezi charakteristické vlastnosti tohoto systému patří velmi minimalistická knihovna, volitelné úložiště (in-memory, on-disc nebo hybridní), široká škálovatelnost a vysoká míra multiplatformnosti díky implementaci v nativním jazyce C. Díky tomu systém eXtremeDB našel uplatnění i v zařízení jako DVD přehrávače, set-top boxy, výrobní a průmyslové řídící systémy a jiné. Systém nabízí přístup definovaným API rozhraním i verzi s podporou SQL jazyka.

Dalším systémem společnosti McObject je systém Perst[23]. Jedná se o embedded SŘBD, který je objektově orientovaný, pro jazyky Java a C#. Oba systémy jsou licencovány pro komerční užití. ExtremeDB i Perst implementují datové struktury sekvenční pole, B-strom, R-strom, hašovací tabulka.

3.7 IBM DB2

Historie IBM DB2[24] sahá až k počátkům sedmdesátých let minulého století, kdy Edgar F. Codd, výzkumný pracovník společnosti IBM, popsal teorii relačních databází[25] a v červnu roku 1970 publikoval model pro manipulaci s daty. V roce 1974 výzkumné středisko IBM v San Jose vyvinulo relační SŘBD nazvaný System R[26], který implementoval Coddovy postupy. Systém DB2 se poprvé objevil v letech 1983.

Systém je multiplatformní s podporou 32bit i 64bit verze operačních systému Linux, Microsoft Windows, z/OS a další. IBM DB2 implementuje datové struktury sekvenční pole a B-strom.

3.8 Souhrn

Pro testování bylo voleno z embedded databázových systémů, které nabízejí přímou podporu jazyka C# nebo jednoduchou implementaci rozhraní jazyka C++, viz tabulka č. 1.

	Sekvenční pole	B-strom	R-strom	Hašovací tabulka
Firebird SQL	✔	✔	✖	✖
MySQL SQL	✔	✔	✖	✔
SQL CE SQL	✔	✔	✖	✖
SQLite SQL	✔	✔	✔	✖
RadegastDB SQL	✔	✔	✔	✔

Tabulka 1: Testované embedded SŘDB a jejich datové struktury

4 Analýza testovacího rozhraní

4.1 Vize

Testovací rozhraní bude určeno pro testera embedded SŘBD. Smyslem rozhraní bude porovnání jednotlivých embedded SŘBD a datových struktur s uložením výsledků pro následné vyhodnocení. Pro spuštění testu budou důležitá vstupní konfigurační data uživatele. Ta se budou skládat ze zvolené kolekce, datové struktury a výběru eDB s jejich nastavením. Výstupem systému bude sumář akcí testu jako vkládání a vyhledávání ve zvolených eDB s vyjádřením propustnosti a časové náročnosti.

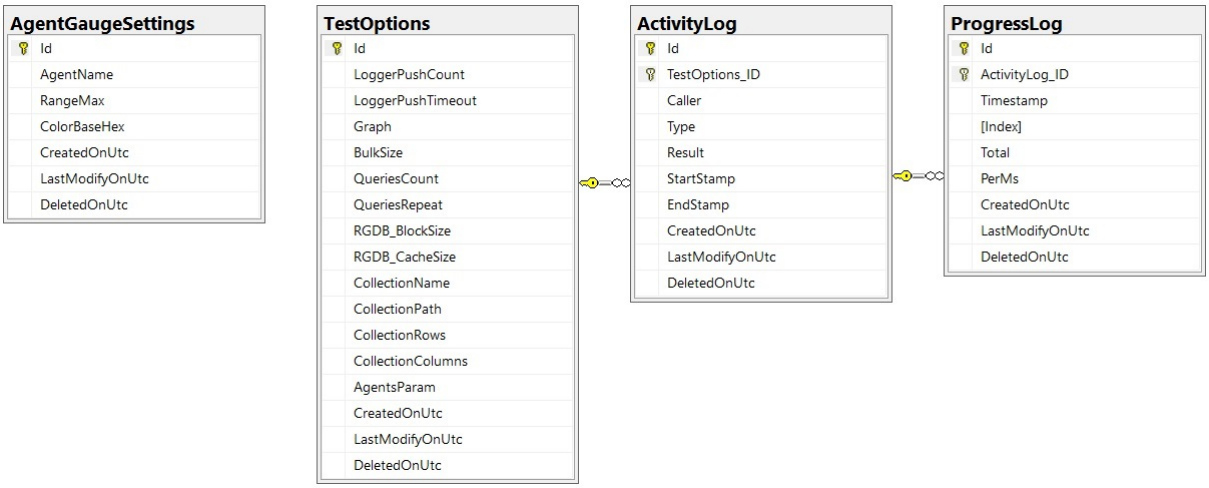
Událost	Reakce
Spuštění testu	Konfigurace eDB, sekvenční spuštění testu, zobrazení průběhu, uložení výsledků
Vyhodnocení testu	Zobrazení sumáře výsledků formou tabulky
Nastavení systému	Nastavení jednotlivých barev a rozsahu průběhu testu eDB

Tabulka 2: Události a reakce testovacího rozhraní

4.2 Datová analýza

V datové analýzy byly navrženy tabulky pro uchovávání výsledků testů a nastavení systému.

4.2.1 ER diagram



Obrázek 8: ER diagram

4.2.2 Datový slovník

Legenda pro popis schéma tabulky: **Primární klíč**; Cizí klíč; *povoluje hodnotu null*

Atribut	Typ	Popis
Id	int	Primární klíč
AgentName	text	Název agenta
RangeMax	int	Maximální rozsah budíku
ColorBaseHex	text	Barva v hexa kódu
CreatedOnUtc	datetime	Datum a čas vytvoření
LastModifyOnUtc	datetime	Datum a čas editace
DeletedOnUtc	datetime	Datum a čas odstranění

Tabulka 3: Schéma tabulky AgentGaugeSettings

Atribut	Typ	Popis
Id	int	Primární klíč
LoggerPushCount	int	Logování po x akcích
LoggerPushTimeout	int	Logování po x milisekundách
Graph	int	Typ datové struktury
BulkSize	int	Vkládaných řádků najednou
QueriesCount	int	Počet dotazů
QueriesRepeat	int	Počet opakování dotazů
RGDB_BlockSize	int	RadegastDB velikost bloku
RGDB_CacheSize	int	RadegastDB velikost cache
CollectionName	varchar(64)	Název kolekce
CollectionPath	text	Cesta ke kolekci
CollectionRows	int	Počet řádků kolekce
CollectionColumns	int	Počet sloupců kolekce
AgentsParam	text	Seznam použitých eDB
CreatedOnUtc	datetime	Datum a čas vytvoření
LastModifyOnUtc	datetime	Datum a čas editace
DeletedOnUtc	datetime	Datum a čas odstranění

Tabulka 4: Schéma tabulky TestOptions

Atribut	Typ	Popis
Id	int	Primární klíč
TestOptions_ID	int	Cizí klíč na možnosti testu
Caller	text	Vlastník
Type	int	Typ akce
Result	int	Typ výsledku
StartStamp	bigint	Časová známka zahájení
EndStamp	bigint	Časová známka ukončení
CreatedOnUtc	datetime	Datum a čas vytvoření
LastModifyOnUtc	datetime	Datum a čas editace
DeletedOnUtc	datetime	Datum a čas odstranění

Tabulka 5: Schéma tabulky ActivityLog

Atribut	Typ	Popis
Id	int	Primární klíč
ActivityLog_ID	int	Cizí klíč na aktivitu
Timestamp	int	Časová známka
Index	text	Pořadí
Total	text	Celkový počet
PerMs	text	Rychlost (operací/ms)
CreatedOnUtc	int	Datum a čas vytvoření
LastModifyOnUtc	datetime	Datum a čas editace
DeletedOnUtc	datetime	Datum a čas odstranění

Tabulka 6: Schéma tabulky ProgressLog

4.3 Funkční analýza

Funkce prováděné systémem

- Nastavení systému
- Spuštění testu
- Vyhodnocení testů
  - Seznam testů
  - Detail testu

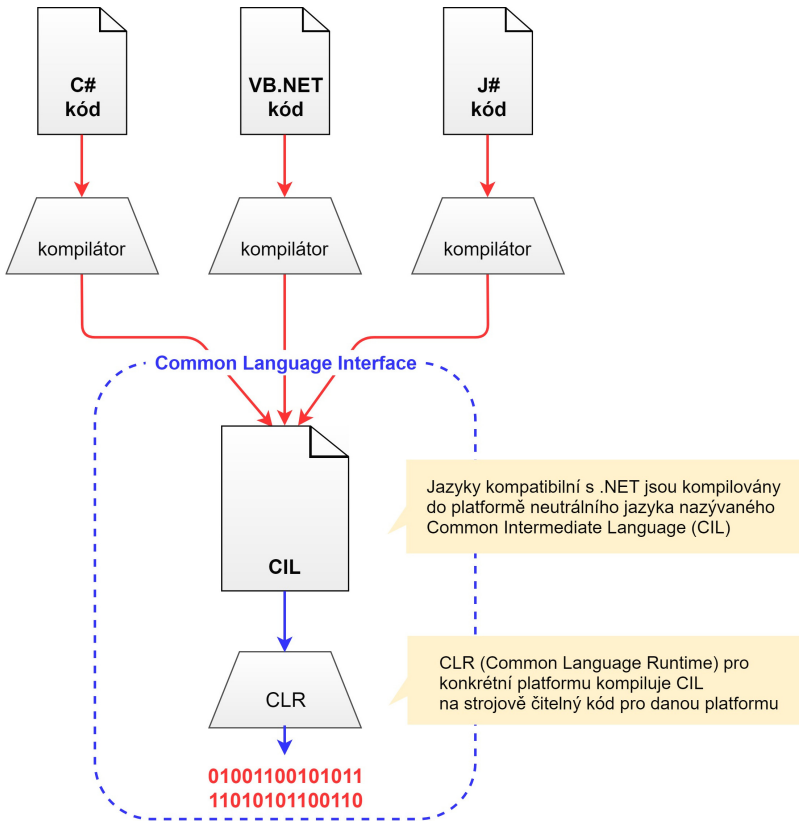
5 Použité technologie

5.1 Aplikační vrstva

Jako aplikační vrstva byly zvoleny nejnovější dostupné technologie .NET Frameworku[27].

5.1.1 .NET Framework

Microsoft začal na vývoji .NET Frameworku[27] koncem devadesátých let, původně pod názvem NGWS (z anglického Next Generation Windows Services). V roce 2000 pracovaly společnosti Microsoft, Hewlett-Packard a Intel na standardizaci CLI (z anglického Common Language Infrastructure)[82] a jazyka C#. Později roku 2000 byla uvolněna beta verze .NET Framework 1.0, první běhové prostředí CLR (z anglického Common Language Runtime)[83] pro jazyky CLI. V roce 2001 vydali ECMA standard a v roce 2003 následovala standardizace ISO. Kompilace programu v jazyce CLI až po strojový kód viz. obrázek č. 9.



Obrázek 9: Common Language Infrastructure (CLI)[28]

Součásti .NET Framework:

- webové aplikace - ASP.NET WebForms[84], ASP.NET MVC[33]
- webové služby, komunikační infrastruktura - ASP.NET Web API[85], Windows Communication Foundation (WCF)[86]
- desktopové aplikace - konzole, WinForms[88], Windows Presentation Foundation (WPF)[87]
- mobilní aplikace - Xamarin (Android, iOS, Windows Phone)[89]
- práce s daty - ADO.NET[10], LINQ[37]
- sdílené knihovny - Portable Library Class (PLC)[90], Shared Project[91]

.NET Framework nabízené jazyky se specifikací pro CLI dle The .NET Language Strategy[29]

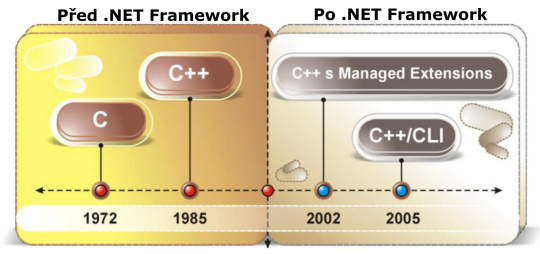
- C#[30] - nejrozšířenější OOP jazyk, je podobný jazykům jako Java, C++, Delphi
- F#[31] - OOP jazyk, především funkcionální
- Visual Basic .NET[32] - jedná se následovníka jazyka Visual Basic

5.1.2 C++/CLI

Počátkem sedmdesátých let dvacátého století vznikl programovací jazyk C[41], který byl navržen a implementován Dennisem Ritchiem a jeho spolupracovníky v Bellových laboratořích. C byl projektován jako kompilovaný jazyk střední úrovně, protože programátorům nabízel mnohem větší úroveň abstrakce od hardwarové struktury než jazyky symbolických instrukcí.

Jazyk C je typickým představitelem programovacího jazyka pro strukturované programování (program strukturovaný jako množina funkcí). O zavedení OOP principů do jazyka C se postaral Bjarne Stroustrup a dal tak vzniku jazyka C++, který byl roku 1998 poprvé standardizován.

Při zavedení .NET Frameworku došlo k zavedení Visual C++.NET, který uvedl jazyk C++ s Managed Extensions (Managed Extensions for C++), který nesl úpravy pro .NET platformu. Koncem roku 2005 .NET Framework 2.0 s sebou uvedl jazyk C++/CLI. Jednalo se o nástupce C++ s Managed Extensions, který odstraňoval předchůdcovy nedostatky. Milníky jazyka C viz. obrázek č. 10.



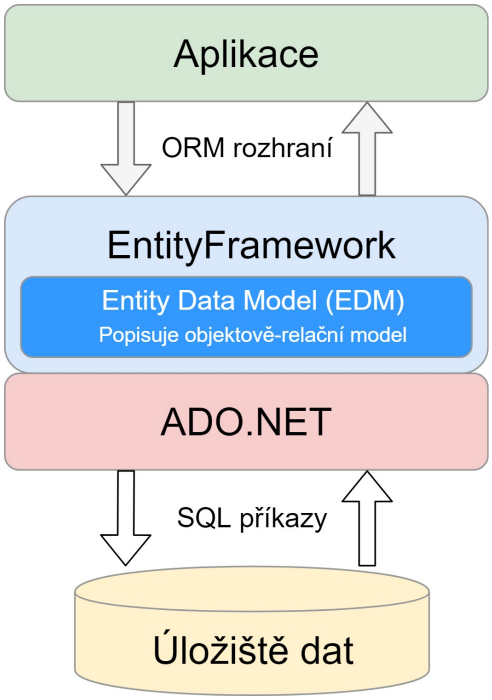
Obrázek 10: Od nativního C po řízené C++/CLI[41]

5.2 Datová vrstva

Pro práci s daty byl zvolen Entity Framework (dále jenom EF)[35]. EF je objektovně relační mapování, které umožňuje .NET vývojářům pracovat s daty pomocí objektů specifických pro danou doménu. Eliminuje množství kódu, které musí vývojář napsat pro standardní CRUD operace nad daty s pomocí ADO.NET, viz. obrázek č. 11.

První verze z roku 2008 byla součástí .NET Framework 3.5 SP1 a Visual Studio 2008 SP1. Tato verze byla značně kritizována a dokonce bylo sepsáno "hlasování o nedůvěře", kterou podepsalo přibližně tisíc programátorů. Druhá verze, nazvána Entity Framework 4.0, jenž byla součástí .NET Frameworku 4.0 z roku 2010, opravovala spoustu kritik z první verze. Třetí verze, Entity Framework 4.1 z roku 2011, podporovala Code First model[36]. Od verze 6.0 je oddělen od .NET Frameworku a vyvíjen jako open source projekt. Pro multiplatformní systémy vznikl v roce 2016 Entity Framework Core 1.0.

Ruku v ruce s EF jde LINQ (z anglického Language Integrated Query)[37]. LINQ je intergrovaný jazyk .NET Frameworku pro dotazování. Hlavním přínosem LINQ je jednotná syntaxe pro přístup k datům bez ohledu na zdroji. Zdrojem tedy může být ADO.NET, XML nebo objekt v paměti. Jako SŘBD byl zvolen SQLite, který má podporu EF6 a LINQ.



Obrázek 11: Entity Framework model[38]

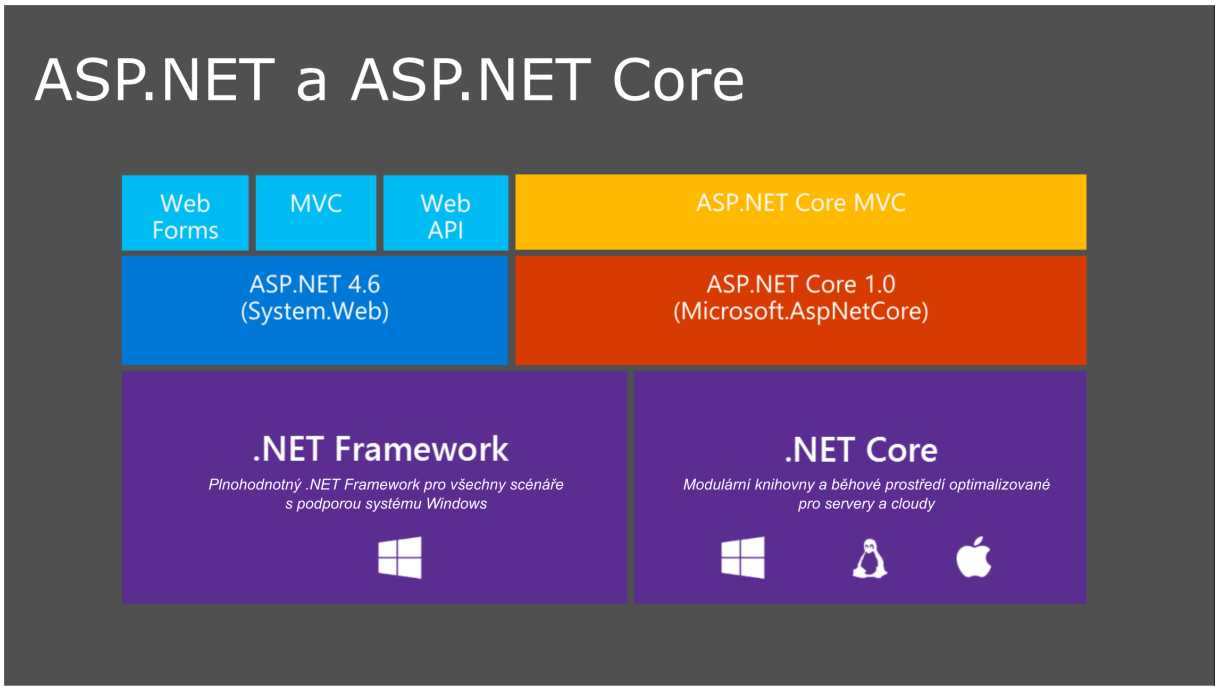
5.3 Prezentační vrstva

Pro prezentační vrstvu byla použita další z .NET Framework[27] technologií - ASP.NET MVC, kterou doplnily technologie HTML5[79], CSS[42] s nadstavbou LESS[43] a JavaScript[77] s nadstavbou TypeScript[74] a Knockout[75].

5.3.1 ASP.NET

ASP.NET[33] je open source aplikační framework na straně serveru, navržen společností Microsoft pro vývoj a správu webových stránek s dynamickým obsahem, webových aplikací a webových služeb. V roce 2001 společně s .NET Frameworkem 1.0 byla spuštěna i první verze toho aplikačního frameworku a jednalo se o nástupce technologie ASP (z anglického Active Server Pages) společnosti Microsoft. ASP.NET je navržen pro CLR a je možné jej vyvíjet s kterýmkoliv jazykem rodiny .NET. Rozšíření ASP.NET SOAP umožňuje komponentě ASP.NET zpracovávat SOAP (z anglického Simple Object Access Protocol)[40] zprávy.

Dalším nástupcem této technologie je ASP.NET Core. Jedná se o reimplementaci ASP.NET, která je společně s Entity Framework Core[20] postavena na nové kompilovací platformě nazývané Roselyn[39] a je multiplatformní. Shrnutí architektur viz. obrázek č.12. Pro vypracování testovacího rozhraní nebyl použit ASP.NET Core, jelikož v době vypracování této práce, byl ve verzi 1.0 a .NET Core nenabízel tak široké možnosti a stabilitu jako .NET Framework.



Obrázek 12: Architektura ASP.NET a ASP.NET Core[34]

5.3.2 MVC

(z anglického Model-View-Controller)[81] je softwarový architektonický vzor, který rozděluje datový model, uživatelské rozhraní a řídící logiku do tří nezávislých komponent. Tak aby změna jedné z nich měla co nejmenší dopad na ostatní.

5.3.3 HTML5

HTML (z anglického HyperText Markup Language)[79] je značkovací jazyk používán pro tvorbu webových stránek. Je to hlavní jazyk pro vytváření dokumentů pro systém World Wide Web (WWW), který umožňuje publikaci dokumentů na internetu. V roce 1989 spolupracovali Tim Berners-Lee a Robert Cailliau na propojeném informačním systému pro CERN, výzkumné centrum fyziky poblíž Ženevy ve Švýcarsku. V té době se pro tvorbu dokumentů obvykle používaly jazyky Tex[71], PostScript[72] a SGML[73]. Berners-Lee si uvědomoval, že pro tvorbu dokumentů potřebují něco jednoduššího a tak v roce 1990 byl navržen HTML a protokol HTTP (z anglického HyperText Transfer Protocol). Zároveň také vznikl první webový prohlížeč World Wide Web.

5.3.4 CSS

CSS neboli kaskádové styly (z anglického Cascading Style Sheets)[42] slouží k definici vzhledu webových stránek v jazyce HTML. Hlavním smyslem CSS je umožnit návrhářům oddělit vzhled dokumentu od jeho struktury a obsahu. Jazyk byl navržen standardizační organizací W3C, autorem prvního návrhu byl Håkon Wium Lie. Jelikož CSS neumožňuje symbolický zápis proměnných nebo konstanty, všechny hodnoty musí být vepsány přímo v kódu. Pokud se tedy na více místech používá stejná barva, musíte ji všude zapisovat ručně namísto využití proměnných. Toto omezení odstraňují preprocesory, např. LESS.

LESS (z anglického Leaner-Style Sheets)[43] je zpětně kompatibilní rozšíření pro CSS. Umožňuje práci s proměnnými, výpočty, funkcemi a další. První verze vznikla v roce 2009, autorem Alexis Sellierem. Původním jazykem implementace byl Ruby, později došlo k portování na JavaScript.

5.3.5 JavaScript

JavaScript[77], obvykle nazýván JS, je interpretovaný programovací jazyk, který spolu s HTML a CSS tvoří základ technologie World Wide Web (WWW). Byl vyvíjen týmem programátorů NetScape pod pracovním názvem Mocha. Oficiálně byl nazván LifeScript, ale po sléze byl přejmenován na JavaScript. První verze Javascriptu pro webové prohlížeče byla spuštěna roku 1995. Skript JavaScriptu se spouští po stažení stránky na straně klienta, na rozdíl např. od jazyků PHP[76] či ASP, které se spouští na straně serveru ještě před stažením klientem.

AJAX - neboli Asynchronous JavaScript and XML, je technologie pro vývoj interaktivních webových stránek, která mění obsah stránek dynamicky na základě asynchronních požadavků na server.

TypeScript [74] - obvykle nazýván TS, jedná se o jazykovou nadstavbu od společnosti Microsoft pro jazyk JavaScript. Nadstavba rozšiřuje jazyk o statické typování, namespace, moduly a další atributy, které známe z OOP. Samotný TypeScript je kompilován do JavaScript kódu.

Knockout [75] - knihovna implementace návrhového vzoru MVVM (z anglického Model-View-ViewModel)[80] v jazyce JavaScript. Knockout je vyvíjen jako open source projekt Stevem Sandersonem, zaměstnancem Microsoft.

6 Implementace testovacího rozhraní

Při implementaci testovacího rozhraní bylo zapotřebí vytvořit třídy pro práci s kolekcí dat, tvorbu SQL příkazů, práci s embedded SŘBD a v neposlední řadě logger, který odesílá uživateli informace o stavu a průběhu testu a stará se o uložení výsledku do databáze.

6.1 Práce s kolekcí dat

Na základě nastavení objekt třídy *EDBTDataBuilder* načítá data ze souboru kolekce nebo kolekci generuje. Také se stará o výběr dotazovaných n-tic a je tedy zdrojem dat pro spouštěné testy. Má-li embedded SŘBD nadstavbu pro SQL dialekt jsou pomocí třídy *EDBTSqlQueryBuilder* a jejich potomků (specifických pro každý embedded SŘBD) vygenerovány SQL příkazy pro vytvoření databáze, tabulky, vložení dat a dotazování.

Obrázek 13: Třídní diagram EDBTDataBuilder

6.2 Práce s embedded SŘBD

O práci s embedded SŘBD se stará třída *EDBTAgent*, která definuje operace testu. Inicializaci, spouštění a posloupnoust operací má na starost třída *EDBTTester*.

Prováděné operace na základě nastavení testu jsou vytvoření databáze (souboru), vytvoření tabulky, vytvoření indexu (datové struktury), vložení dat a dotazování se nad daty.

RGDBAgent - obsluhuje RadegastDB pomocí knihovny RadegastDB.Wrapper. Ve skutečnosti se nejedná o wrappovací knihovnu, ale o kompletní knihovnu RadegastDB zkompilovanou v C++/CLI. Umožňuje testování sekvenčního pole, B-stromu, R-stromu a hašovací tabulky.

MySQLAgent - vytváří a řídí instanci MySQL Embedded serveru pomocí importovaných funkcí z knihovny verze 5.7. Umožňuje testování sekvenčního pole, B-stromu a hašovací tabulky.

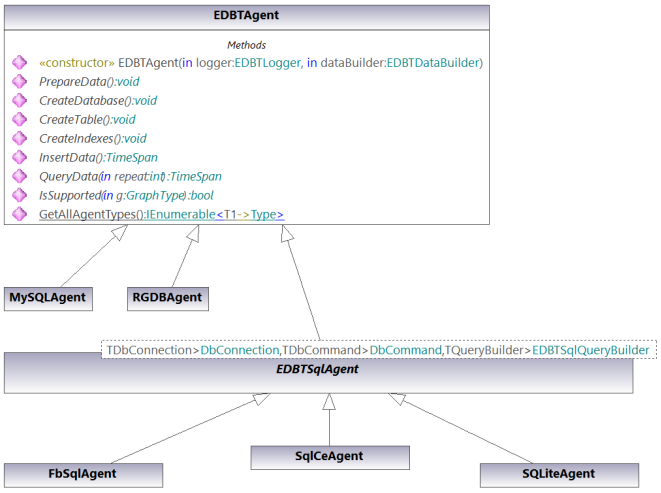
6.2.1 EDBTSqlAgent

Potomek třídy EDBTAgent, který definuje pro operace testu spouštění dotazů pomocí ADO.NET.

FbSqlAgent - pomocí knihovny FirebirdSql.Data.FirebirdClient řídí SŘBD Firebird verze 2.5. Umožňuje testování sekvenčního pole, B-stromu.

SqlCeAgent - s použitím knihovny System.Data.SqlServerCe řídí systém SqlCe 4.0. Umožňuje testování sekvenčního pole, B-stromu.

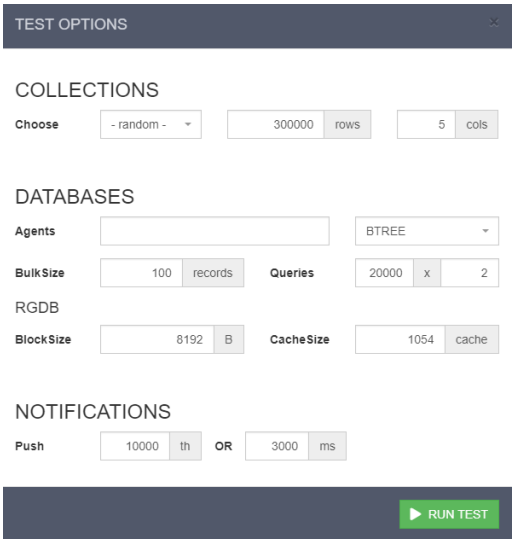
SQLiteAgent - pomocí knihovny System.Data.SQLite řídí SŘBD SQLite verze 3.0. Umožňuje testování sekvenčního pole, B-stromu a R-stromu.



Obrázek 14: Třídní diagram agentů

6.3 Vizualizace průběhu

V rámci konfigurace testu je zapotřebí zvolit kolekci, testované eDB s jejich nastavením a specifikovat datovou strukturu, viz. obrázek č. 15. Systém aktuálně pracuje s kolekcemi POKER (1 000 000 x 11)[1] a TIGER[2] (5 889 786 x 3).



Obrázek 15: Rozhraní konfigurace testu

Vizualizace průběhu probíhá pomocí několika informativních prvků za pomocí loggeru třídy *EDBTLogger*. Ten pošle pomocí SignalR notifikaci klientskému prohlížeči. Klientský prohlížeč notifikaci zpracuje a JavaScriptem aktualizuje informativní prvky rozhraní, viz. obrázek č. 16.

Obrázek 16: Rozhraní vizualizace spuštěného testu

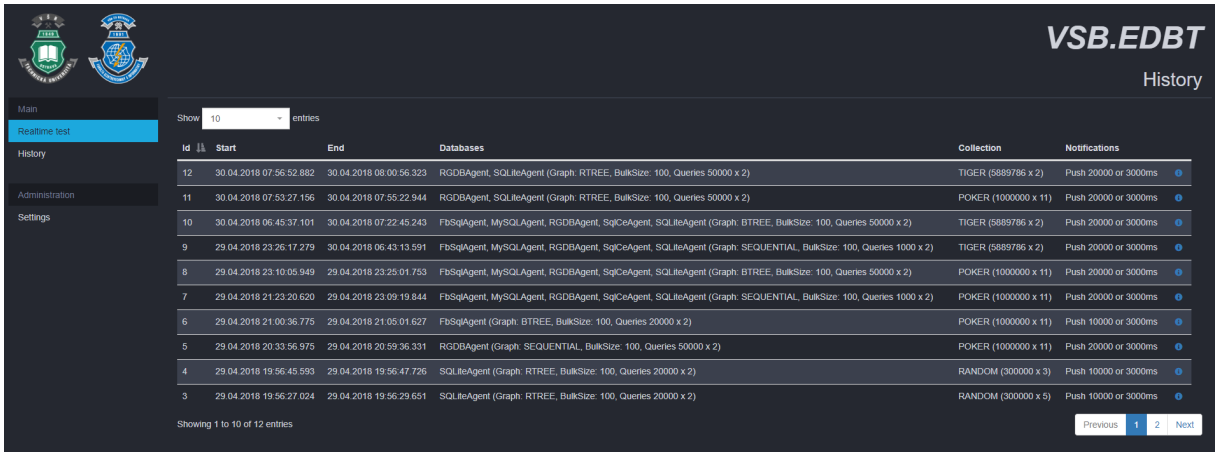
[1] http://www.census.gov/geo/www/tiger/
[2] http://archive.ics.uci.edu/ml/datasets/Poker+Hand

Popis rozhraní:

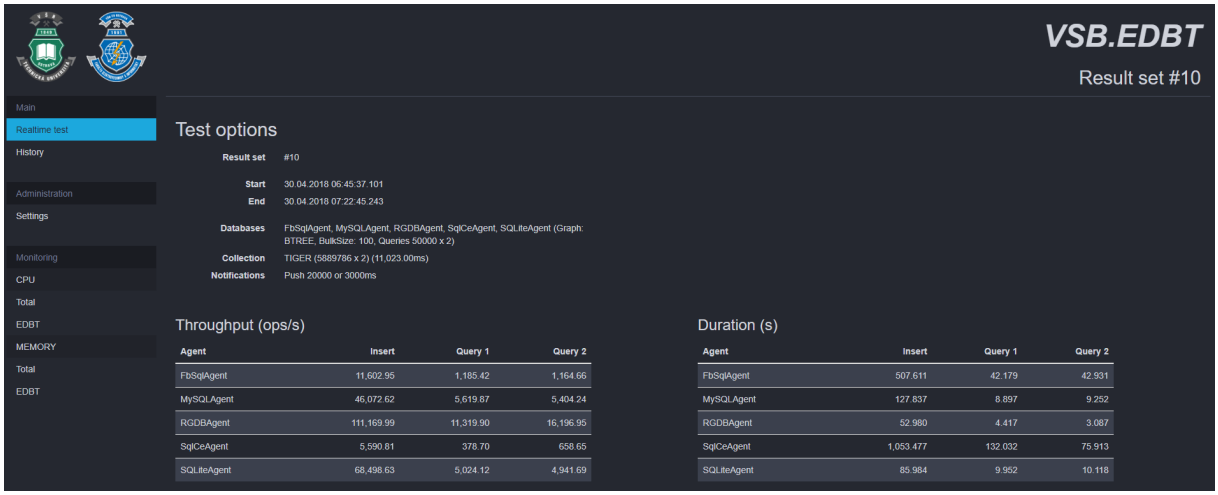
1. Nastavení testu
2. Textové vyjádření průběhu
3. Grafické procentuální vyjádření průběhu
4. Rychloměr propustnosti vkládání a vyhledávání
5. Graf průběhu propustnosti všech operací
6. Textový log přijatých notifikací

6.4 Vyhodnocení testů

Do interní databáze testovacího rozhraní se ukládá konfigurace testu i průběh. Poté je možné si od kteréhokoli testu zobrazit vyhodnocení vyjádřené v podobě tabulky propustnosti a časové náročnosti. Viz. obrázek č. 17 a č. 18.



Obrázek 17: Rozhraní přehledu spuštěných testů



Obrázek 18: Rozhraní detailu testu s vyhodnocením

7 Porovnání výkonů datových struktur

Pro všechny testy byly zvoleny stejné datové kolekce s rozdílnou datovou strukturu a počtem dotazů. Použité kolekce byly POKER a TIGER. Vkládání záznamů proběhlo s maximálním počtem 100 záznamů najednou. Následující tabulky zobrazují propustnost počtu operací za sekundu.

7.1 Sekvenční pole

Pro sekvenční pole bylo zvoleno 1000 dotazů s počtem opakování 2.

	Vkládání [op./s]	Dotazování 1 [op./s]	Dotazování 2 [op./s]
FirebirdSQL	5 166,28	0,80	0,71
MySQL	182 915,68	5,27	5,19
RadegastDB	372 717,11	34,65	33,01
SQL CE	4 851,90	0,82	0,82
SQLite	51 591,60	5,50	5,40

Tabulka 7: Vyhodnocení datové struktury sekvenční pole a kolekce POKER

	Vkládání [op./s]	Dotazování 1 [op./s]	Dotazování 2 [op./s]
FirebirdSQL	15 166,73	0,19	0,19
MySQL	413 434,37	1,42	1,42
RadegastDB	392 600,05	8,88	8,84
SQL CE	6 798,07	0,18	0,18
SQLite	154 259,61	1,23	1,23

Tabulka 8: Vyhodnocení datové struktury sekvenční pole a kolekce TIGER

7.2 B-strom

Pro testování B-stromu bylo zvoleno 50 000 dotazů s počtem opakování 2.

	Vkládání [op./s]	Dotazování 1 [op./s]	Dotazování 2 [op./s]
FirebirdSQL	4 919,18	1 309,17	1 287,86
MySQL	23 523,32	4 107,11	3 901,37
RadegastDB	68 226,79	18 642,80	22 593,76
SQL CE	3 387,74	624,51	872,39
SQLite	22 264,77	4 968,70	4 874,24

Tabulka 9: Vyhodnocení datové struktury B-strom a kolekce POKER

	Vkládání [op./s]	Dotazování 1 [op./s]	Dotazování 2 [op./s]
FirebirdSQL	11 602,95	1 185,42	1 164,66
MySQL	46 072,62	5 619,87	5 404,24
RadegastDB	111 169,99	11 319,90	16 196,95
SQL CE	5 590,81	378,70	658,65
SQLite	68 498,63	5 024,12	4 941,69

Tabulka 10: Vyhodnocení datové struktury B-strom a kolekce TIGER

7.3 R-strom

Jelikož SQLite má R-strom navržený pro 3 až 11 sloupců, kde první je vždy Id a další jsou dvojice minimum a maximum, nebylo jej možné se zvolenými kolekcemi testovat. Pro testování bylo vytvořeno speciální generování kolekce. Kolekce GEN1 s 1 000 000 řádků dimenze 11 a kolekce GEN2 s 5 000 000 řádky a dimenzí 3. Spuštěno bylo 50 000 dotazů s počtem opakování 2.

	Vkládání [op./s]	Dotazování 1 [op./s]	Dotazování 2 [op./s]
RadegastDB	41 447,34	8 099,79	8 168,60
SQLite	14 602,59	4 234,06	4 133,26

Tabulka 11: Vyhodnocení datové struktury R-strom a kolekce GEN1

	Vkládání [op./s]	Dotazování 1 [op./s]	Dotazování 2 [op./s]
RadegastDB	41 843,81	7 080,15	8 986,34
SQLite	18 781,25	2 024,05	3 817,96

Tabulka 12: Vyhodnocení datové struktury R-strom a kolekce GEN2

7.4 Hašovací tabulka

Vzhledem k tomu, že většina testovaných embedded SŘBD nemá podporu pro hašovací tabulku, rozhodl jsem se tuto datovou strukturu netestovat.

8 Závěr

Při vyhodnocení testů je zřejmé, že embedded SŘBD FirebirdSQL a SQL CE nejsou zcela vhodné pro tento typ testování. Jejich pomalé vkládání i vyhledávání rapidně prodlužuje dobu trvání testu. Přijatelnější by pro ně bylo testování s kolekcí o menším počtu záznamů.

Na druhou stranu MySQL, RadegastDB a SQLite si vedly v testu velice dobře. Nejrychlejší je vkládání do sekvenčního pole kde tyto tři systémy dosahují opravdu vysokých propustností. Ovšem vyhledávání je tomu pravým opakem. Nejrozumnější pro rychlé vkládání i vyhledávání se jeví B-strom a to je důvod proč se jedná o nejpoužívanější datovou strukturu. Datová struktura R-stromu je nejužitečnější v případě souřadnicových systémů.

RadegastDB byl na tom ve všech testech s propustností lépe než ostatní systémy. Nutno ale podotknout, že RadegastDB v současné době nenabízí možnost ovládání pomocí SQL dialektu, které by zabralo strojový čas a tím snížilo jeho náskok.

Literatura

[1] CHOVANEC Petr, Ing. *Reduction of Disk accesses in multidimensional data structures.* 2015 [cit. 2018-04-10] Dizertační práce, Vysoká škola Báňská-Technická Univerzita Ostrava

[2] BAČA R., CHOVANEC P., KRÁTKÝ m. a LUKÁŠ P. *QuickDB - Yet Another Database Management System?.* 2014 [cit. 2018-04-10]. Dateso

[3] BAYER, R. a E. MCCREIGHT. Organization and maintenance of large ordered indices. In: *Proceedings of the 1970 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control - SIGFIDET '70* [online]. New York, New York, USA: ACM Press, 1970, 1970, s. 107- [cit. 2018-04-22]. DOI: 10.1145/1734663.1734671. <http://portal.acm.org/citation.cfm?doid=1734663.1734671>

[4] POKORNÝ Jan, *Prostorové datové struktury a jejich použití k indexaci prostorových objektů* [online]. c2000 [cit. 2018-04-14]. <http://gis.vsb.cz/GIS_Ostrava/GIS_Ova_2000/Sbornik/Pokorny/Referat.htm>

[5] GUTTMAN, Antonin. R-trees. *ACM SIGMOD Record* [online]. 1984, 14(2), 47- [cit. 2018-04-22]. DOI: 10.1145/971697.602266. ISSN 01635808. <http://portal.acm.org/citation.cfm?doid=971697.602266>

[6] SEDGEWICK, Robert. *Algorithms in Java.* 3rd ed. Boston: Addison-Wesley, 2004. ISBN 0-201-36120-5.

[7] SEDGEWICK, Robert. *Algorithms in Java.* 3rd ed. Boston: Addison-Wesley, 2004. ISBN 0-201-36120-5. Kapitola 14.3: Linear Probing.

[8] SEDGEWICK, Robert. *Algorithms in Java.* 3rd ed. Boston: Addison-Wesley, 2004. ISBN 0-201-36120-5. Kapitola 14.4: Double hashing.

[9] ORACLE. *Oracle Database* [online]. c2018 [cit. 2018-04-14]. <https://www.oracle.com/cz/database/index.html>

[10] MICROSOFT. *ADO.NET* [online]. c2018 [cit. 2018-04-14]. <https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/index>

[11] FIREBIRD PROJECT. *Firebird 2.5 Language Reference* [online]. c2017 [cit. 2018-04-14]. <https://www.firebirdsql.org/en/documentation>

[12] EMBARCADERO TECHNOLOGIES. *InterBase* [online]. c2018 [cit. 2018-04-14]. <https://www.embarcadero.com/products/interbase>

[13] ORACLE. *MySQL* [online]. c2018 [cit. 2018-04-14]. <https://www.mysql.com>

[14] ORACLE. *MySQL - libmysqld, the Embedded MySQL Server Library* [online]. c2018 [cit. 2018-04-14]. <https://dev.mysql.com/doc/refman/5.7/en/libmysqld.html>

[15] MARIADB FOUNDATION. *MariaDB - About* [online]. c2018 [cit. 2018-04-14]. <https://mariadb.org>

[16] MICROSOFT. *Microsoft SQL Server Compact 4.0 Books Online* [online]. c2011 [cit. 2018-04-14]. <https://www.microsoft.com/en-us/download/details.aspx?id=21880>

[17] MICROSOFT. *Microsoft SQL Server 2017* [online]. c2018 [cit. 2018-04-14]. <https://www.microsoft.com/cs-cz/sql-server/sql-server-2017>

[18] SQLITE. *Documentation* [online]. c2018 [cit. 2018-04-14]. https://www.sqlite.org/docs.html, 2018.

[19] GNU. *GDBM* [online]. c2018 [cit. 2018-04-14]. <https://www.gnu.org.ua/software/gdbm>

[20] MICROSOFT. *Entity Framework Core Documentation* [online]. c2018 [cit. 2018-04-14]. <https://docs.microsoft.com/en-us/ef/core>

[21] ORACLE. *Oracle - Berkeley DB* [online]. c2018 [cit. 2018-04-14]. <http://www.oracle.com/technetwork/database/database-technologies/berkeleydb/overview/index.html>

[22] MCOBJECT. *eXtremeDB® real-time embedded database* [online]. c2018 [cit. 2018-04-14]. <http://www.mcobject.com/extremedbfamily.shtml>

[23] MCOBJECT. *Perst - An open source, object-oriented embedded database* [online]. c2018 [cit. 2018-04-14]. <http://www.mcobject.com/perst>

[24] IBM. *IBM Db2 Database* [online]. c2018 [cit. 2018-04-15]. <https://www.ibm.com/analytics/us/en/db2>

[25] CODD E. F., *Relational Completeness of Data Base Sublanguages*, Prentice-Hall, 1970.

[26] CHAMBERLIN, Donald D., Franco PUTZOLU, Patricia Griffiths SELINGER, et al., *A History and Evolution of System R*, Communications of the ACM, 1981.

[27] MICROSOFT. *Overview of the .NET Framework* [online]. c2018 [cit. 2018-04-15]. <https://msdn.microsoft.com/en-us/library/a4t23ktk(v=vs.100).aspx>

[28] MICROSOFT, *.NET Framwork* [online]. c2018 [cit. 2018-04-15]. <https://en.wikipedia.org/wiki/Common_Language_Infrastructure>

[29] .NET BLOG. *The .NET Language Strategy* [online]. c2017 [cit. 2018-04-15] <https://blogs.msdn.microsoft.com/dotnet/2017/02/01/the-net-language-strategy>

[30] MICROSOFT. *C# Guige* [online]. c2018 [cit. 2018-04-15] <https://docs.microsoft.com/cs-cz/dotnet/csharp>

[31] MICROSOFT. *F# Guige* [online]. c2018 [cit. 2018-04-15] <https://docs.microsoft.com/cs-cz/dotnet/fsharp>

[32] MICROSOFT. *Visual Basic Guige* [online]. c2018 [cit. 2018-04-15] <https://docs.microsoft.com/cs-cz/dotnet/visual-basic>

[33] MICROSOFT. *ASP.NET Overview* [online], c2018 [cit. 2018-04-15] <https://docs.microsoft.com/en-us/aspnet/overview>

[34] CODE BURST. *ASP.NET Core in a Nutshell* [online]. c2017 [cit. 2018-04-15] <https://codeburst.io/what-you-need-to-know-about-asp-net-core-30fec1d33d78>

[35] MICROSOFT. *Entity Framework*, [online], c2016 [cit. 2018-04-15]. <https://docs.microsoft.com/en-us/ef>

[36] MICROSOFT. *Entity Framework Code First Conventions* [online]. c2016 [cit. 2018-04-15]. <https://msdn.microsoft.com/en-us/library/jj679962(v=vs.113).aspx>

[37] MICROSOFT. *LINQ: .NET Language-Integrated Query* [online]. c2007 [cit. 2018-04-15]. <https://msdn.microsoft.com/en-us/library/bb308959.aspx>

[38] CODE PROJECT. *An Introduction to Entity Framework for Absolute Beginners* [online]. c2012 [cit. 2018-04-15] <https://www.codeproject.com/Articles/363040/An-Introduction-to-Entity-Framework-for-Absolute-B>

[39] MICROSOFT. *The .NET Compiler Platform SDK*. c2018 [cit. 2018-04-15]. <https://docs.microsoft.com/en-us/dotnet/csharp/roslyn-sdk>

[40] W3C. *Simple Object Access Protocol* [online]. c2004 [cit. 2018-04-15]. <https://www.w3.org/TR/soap>

[41] MICROSOFT. *C++/CLI - začínáme programovat* [online]. c2009 [cit. 2018-04-15]. <https://www.w3.org/TR/soap>

[42] W3C. *HTML & CSS* [online]. c2016, [cit.2018-04-15]. <https://www.w3.org/standards/webdesign/htmlcss>

[43] THE CORE LESS TEAM, *LESS* [online]. c2018 [cit. 2018-04-15]. <http://www.lesscss.org>

[44] WIKIPEDIA. *Wikipedia* [online]. c2018 [cit. 2018-04-22]. <http://www.wikipedia.org>

[45] WORDPRESS.COM. *Wordpress.com* [online]. c2018 [cit. 2018-04-22]. <http://www.wordpress.com>

[46] GOOGLE. *Google* [online]. c2018 [cit. 2018-04-22]. <http://www.google.com>

[47] GOOGLE. *Chrome* [online]. c2018 [cit. 2018-04-22]. <http://chrome.google.com>

[48] MOZILLA. *Firefox* [online]. c2018 [cit. 2018-04-22]. <http://www.mozilla.org>

[49] OPERA SOFTWARE. *Opera* [online]. c2018 [cit. 2018-04-22]. <http://opera.com>

[50] DJANGO SOFTWARE FOUNDATION. *Django* [online]. c2018 [cit. 2018-04-22]. <http://www.djangoproject.com>

[51] BUYTAERT Dries. *Drupal* [online]. c2018 [cit. 2018-04-22]. <http://www.drupal.org>

[52] RUBY ON RAILS. *Ruby on Rails* [online]. c2018 [cit 2018-04-21]. <http://www.rubyonrails.org>

[53] BLACKBERRY LIMITED. *Blackberry* [online]. c2018 [cit 2018-04-21]. <http://www.blackberry.com>

[54] NOKIA. *Symbian* [online]. c2018 [cit 2018-04-21]. <http://www.symbian.org>

[55] NOKIA. *Maemo* [online]. c2018 [cit 2018-04-21]. <http://www.maemo.org>

[56] GOOGLE. *Android* [online]. c2018 [cit 2018-04-21]. <http://www.android.com>

[57] LG. *LG SMART TV WEBOS 2.0* [online]. c2018 [cit 2018-04-21]. <http://www.lg.com/cz/webos>

[58] THE NETBSD FOUNDATION. *Welcome to NetBSD* [online]. c2018 [cit 2018-04-21]. <http://www.netbsd.org>

[59] APPLE. *iOS 11* [online]. c2018 [cit 2018-04-21]. <https://www.apple.com/ios/ios-11/>

[60] MICROSOFT. *Windows* [online]. c2018 [cit 2018-04-20]. <https://www.microsoft.com/en-US/windows>

[61] MCKUSICK, Marshall Kirk. *The design and implementation of the 4.4BSD operating system.* Reading, Mass.: Addison-Wesley, c1996. UXIX and open systems series. ISBN 978-0-201-54979-9.

[62] HOWES Tim. *The Lightweight Directory Access Protocol: X.500 Lite.* Poslední revize 26.12.2012 [cit. 2018-04-20]. <https://www.openldap.org/pub/umich/ldap.pdf>

[63] RED HAT, INC. *389 Directory Server* [online]. c2018 [cit 2018-04-20]. <http://directory.fedoraproject.org/

[64] BITCOIN PROJECT. *Bitcoin* [online]. c2018 [cit 2018-04-20]. <http://www.bitcoin.org>

[65] APACHE SOFTWARE FOUNDATION. *Apache Subversion* [online]. c2018 [cit 2018-04-20]. <https://subversion.apache.org>

[66] RAYMOND S.E. *BOGOFILTER v1.2.4* [online]. c2018 [cit 2018-04-17]. <http://bogofilter.sourceforge.net>

[67] APACHE SOFTWARE FOUNDATION. *Apache SpamAssassin* [online]. c2018 [cit 2018-04-20]. <https://spamassassin.apache.org>

[68] *NoSQL* [online]. c2018 [cit 2018-04-20]. <http://nosql-database.org>

[69] ORACLE. *Oracle NoSQL Database* [online]. c2018 [cit 2018-04-20]. <http://www.oracle.com/technetwork/database/database-technologies/nosqldb/overview/index.html>

[70] PROJECT VOLDEMORT. *A distributed database* [online]. c2018 [cit 2018-04-19]. <http://www.project-voldemort.com>

[71] LATEX PROJECT.*An introduction to LaTeX*. c2016 [cit. 2018-04-19]. <https://www.latex-project.org/about>

[72] ADOBE. *Adobe PostScript* [online]. c2018 [cit. 2018-04-18]. <https://www.adobe.com/products/postscript.html>

[73] WORLD WIDE WEB CONSORTIUM (W3C). *Overview of SGML Resources* [online]. c1995, poslední revize 26.03.2004 [cit. 2018-04-18]. <https://www.w3.org/MarkUp/SGML>

[74] MICRSOFT. *TypeScript* [online]. c2018 [cit. 2018-04-18]. <https://www.typescriptlang.org>

[75] KNOCKOUT. *Knockout* [online]. c2018 [cit. 2018-04-18]. <http://knockoutjs.com>

[76] THE PHP GROUP. *PHP* [online]. c2018 [cit. 2018-04-18]. <http://php.net>

[77] MOZILLA. *JavaScript* [online]. c2018 [cit. 2018-04-18]. https://developer.mozilla.org/en-US/docs/Web/JavaScript

[78] SWARTZ A. *A Brief History of Ajax*. Vydáno 22.12.2005 [cit. 2018-04-18].

[79] WORLD WIDE WEB CONSORTIUM (W3C). *HTML* [online]. c2018 [cit. 2018-04-18]. <https://www.w3.org/html>

[80] MICROSOFT. *The MVVM Pattern* [online]. c2018 [cit. 2018-04-18]. <https://msdn.microsoft.com/en-us/library/hh848246.aspx>

[81] MICROSOFT. *Model-View-Controller* [online]. c2018 [cit. 2018-04-18]. <https://msdn.microsoft.com/en-us/library/ff649643.aspx>

[82] ECMA INTRNATIONAL. *Standard ECMA-335: Common Language Infrastructure (CLI)* [online]. Poslední úprava 03.2012 [cit. 2018-04-18]. <https://www.ecma-international.org/publications/standards/Ecma-335.htm>

[83] MICROSOFT. *Common Language Runtime (CLR)* [online]. c2018 [cit. 2018-04-18]. <https://docs.microsoft.com/en-US/dotnet/standard/clr>

[84] MICROSOFT. *ASP.NET Web Forms* [online]. c2018 [cit. 2018-04-18]. https://www.asp.net/web-forms

[85] MICROSOFT. *ASP.NET Web API* [online]. c2018 [cit. 2018-04-18]. https://www.asp.net/web-api

[86] MICROSOFT. *Windows Communication Foundation* [online]. c2018 [cit. 2018-04-18]. <https://msdn.microsoft.com/en-us/library/ms735119(v=vs.90).aspx>

[87] MICROSOFT. *Windows Presentation Foundation* [online]. c2018 [cit. 2018-04-18]. <https://docs.microsoft.com/en-us/dotnet/framework/wpf>

[88] MICROSOFT. *Windows Forms* [online]. c2018 [cit. 2018-04-18]. <https://docs.microsoft.com/en-us/dotnet/framework/winforms>

[89] MICROSOFT. *Xamarin* [online]. c2018 [cit. 2018-04-18]. <https://www.xamarin.com>

[90] MICROSOFT. *Portable Class Libraries* [online], c2018 [cit. 2018-04-18]. <https://msdn.microsoft.com/cs-cz/library/gg597391(v=vs.100).aspx>

[91] YADAV P.K. *Shared Project : An Impressive Feature of Visual Studio 2015 Preview* [online]. c2014 [cit. 2018-04-18]. <https://www.c-sharpcorner.com/UploadFile/7ca517/shared-project-an-impressive-features-of-visual-studio-201/>

A Příloha na CD

\thesis	zdroje textové části bakalářské práce
\sources	zdroje testovacího rozhraní včetně kolekcí a zdrojů RadegastDB